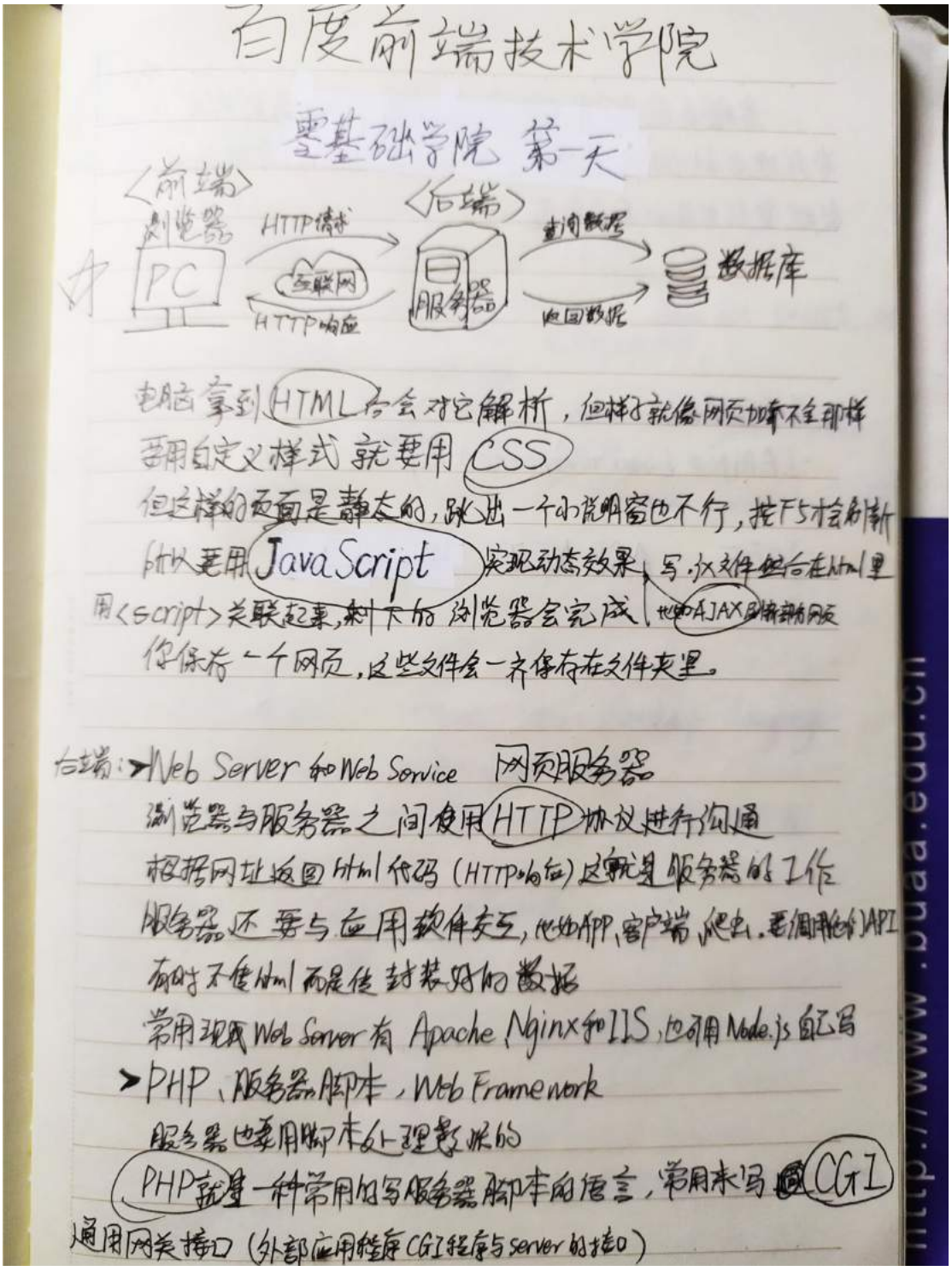
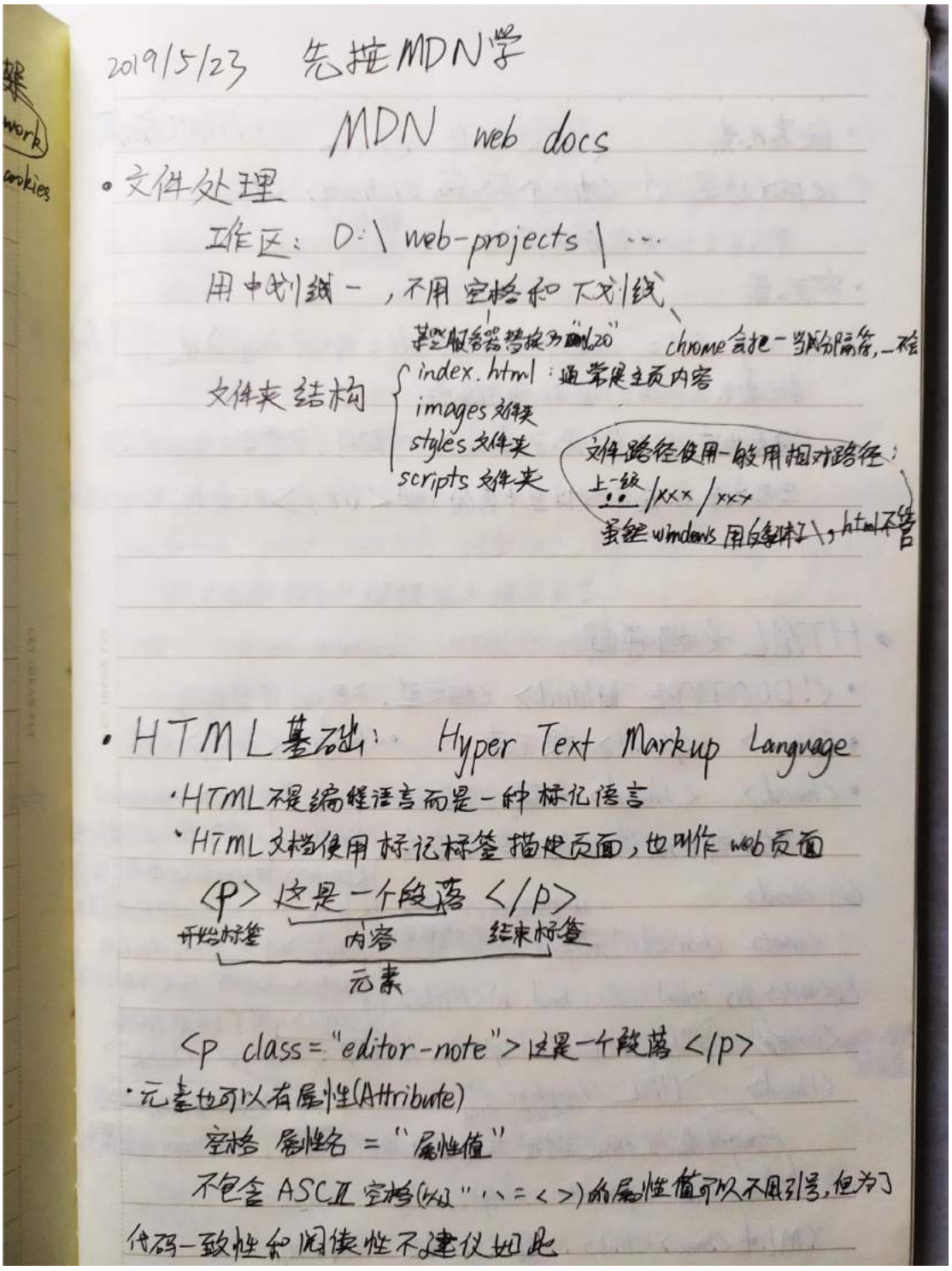
零基础学院第一天、第二天



我们在写服务器脚本的时候，通常还会用个同语言写的 Web Framework 来处理各种细节，防御一些常见的攻击，提供跨站认证（比如用已有的微博账号注册其他网站）的接口，利用 cookie 处理登陆状态和用户设置，生成网页模版之类的。如果你用 C# 或者 Visual Basic 写服务器脚本，就可以用 http://ASP.NET 这个框架实现这些功能，帮你省点麻

烦。不过现在不少人是反过来为了一个好用的 Web Framework 去选择它对应的服务器脚本语言的。



· 嵌套元素　 斜体 　· HTML强调

<p> 这是一个 段落 </p>
粗体

必须层次正确

· 空元素

<img src="images/xxx.png" alt="My test image">　可补全：这是图片（<img>内容）信息

本元素只有开始标签和两个属性

没有内容，因为图像元素嵌入png即可，不需要在后面接文件

空元素应该在开始标签中关闭，如
 定义换行 <img>是自闭合

• HTML文档详解

• <!DOCTYPE html> 文档类型，不需了解保留即可

• <html> </html> 根元素　• <body> </body> 内容

• <head> </head> 头部，包含文档标题等一元信息（meta-information）

常在头部区域添加的元素标签有

- <title>：必须有，浏览器试显示标题且唯一
- <base>：定义所有链接的URL
- <link>：文档与外部资源的关系
- <style>：指定css文件引用
- <meta>：元数据，指定关键词、作者等
- <script>：用于加载脚本

例：<head>

<meta charset="utf-8">　一种字符编码，可以处理绝大多数语言

1. <title> W3Cschool (w3cschool.cn) </title>

2. <base href="https://w3cschool.cn/statics/images/"; target="_blank">

</head>　href → URL (hypertext reference) 超文本引用

target属性规定超链接和表单在何处打开，base中target可被覆盖

有 _blank、_parent、_self、_top、framename

XML中 <base> <link> … 都必须被正确关闭

5/25

3. <link>标签 常用于链接 css 样式表

<link rel="stylesheet" type="text/css" href="styles.css">

属性：charset：定义被链接文档的字符编码方式，html 5 不支持 (char_encoding)

href：URL，被链接文档的位置

hreflang：language_code 文档中文本的语言

media：media_query (询问) 显示在什么设备上

必备属性 rel：alternate、archives (档案)、author、bookmark、external、first、help、icon、last、license、next、nofollow、noreferrer、pingback、prefetch、prev、search、sidebar、stylesheet、tag、up

定义当前文档与被链接文档的关系

rev：reversed relationship HTML5 不支持该属性，功能同 rel

sizes new：Height x Width any 只对 rel="icon" 起作用

target：用法同 base-target，HTML5 不支持

type：MIME_type 什么后缀用什么打开

资源定位符 (相对路径)

4. <meta>标签 元数据，不会显示但可以调用

通常以 名称/值 对出现 <meta name="名称" content="值">

属性：charset (HTML5) content：text 类型，定义与 name 或 http-equiv 相关的信息

http-equiv：content-type default-style refresh (刷新页面用)

用来把 content 属性关联到 HTTP 头部

name：可以是 author、description、keywords (contents 可能多个词) 等...

5. <script>标签

用于链接外部JavaScript文件的外部资源标签
即可包含脚本语句，也可通过src指向外部脚本文件
但若用了"src"，则<script>元素必须是空的

eg1: <script> document.write ("Hello World") </script>

<script src = "myscripts.js"> </script>

HTML4中"type"必需，5中可选

存疑：<frameset> <noscript> CDATA

属性：charset ··

async：异步执行，="async"时页面解析同时脚本执行

defer：="defer"时页面完成后再解析脚本

src：URL

type：MIME-type

xml:space：preserve HTML5不支持

6. <style>标签

例 <style type="text/css">
h1 {color:red;}
p {color:blue;}
</style>
>css

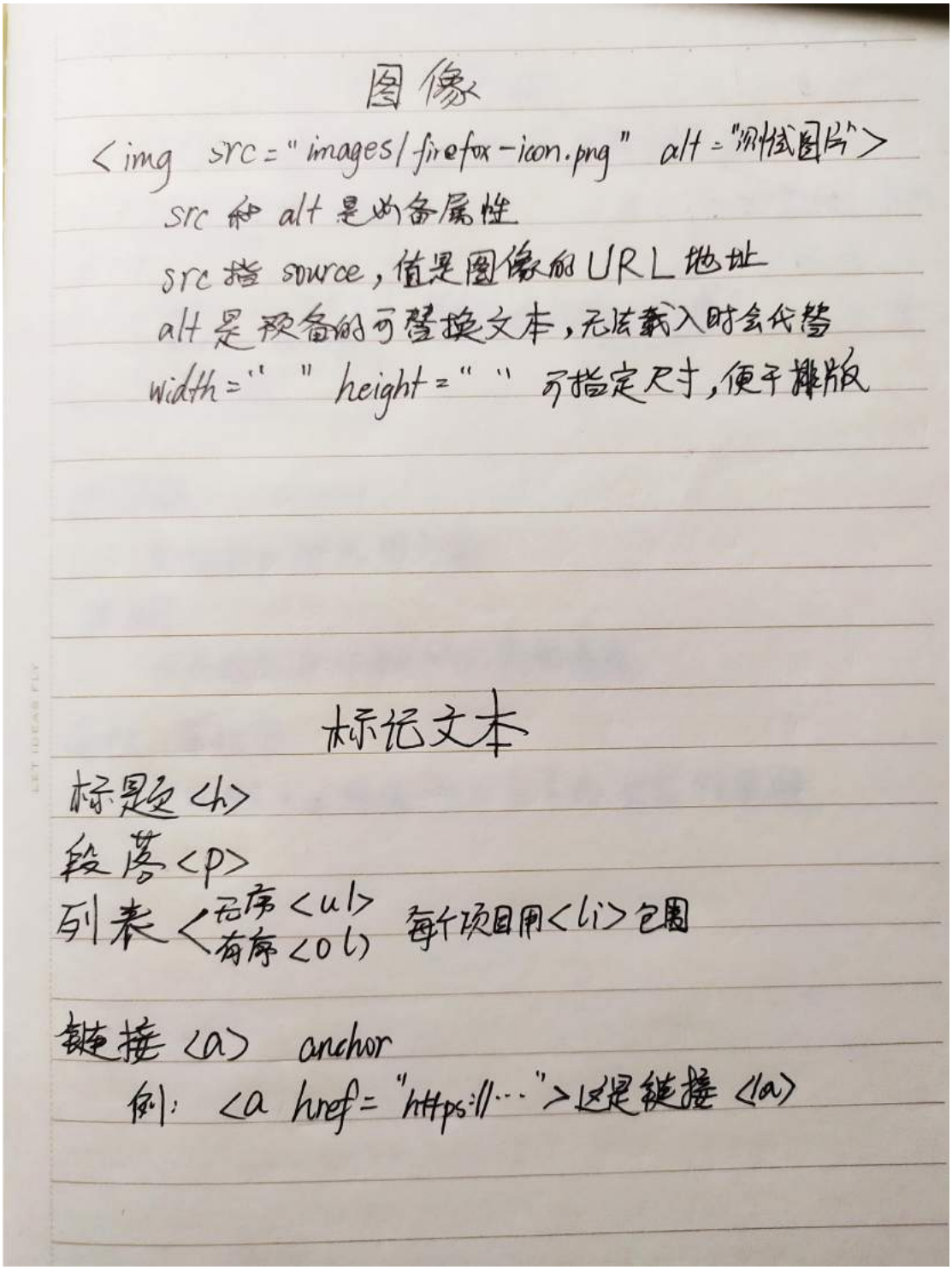
属性：media
~~scoped~~：[scoped] 仅影响<style>的父元素和它的子元素（仅Firefox支持）
type

补充：HTML水平线 <hr>

HTML注释 <!-- 注释 -->

<h1-h6>一般用了级标题，块级元素前后会自动加空行

图像



<img src="images/firefox-icon.png" alt="测试图片">

src 和 alt 是必备属性

src 指 source，值是图像的URL地址

alt 是预备的可替换文本，无法载入时会代替

width="" height="" 可指定尺寸，便于排版

标记文本

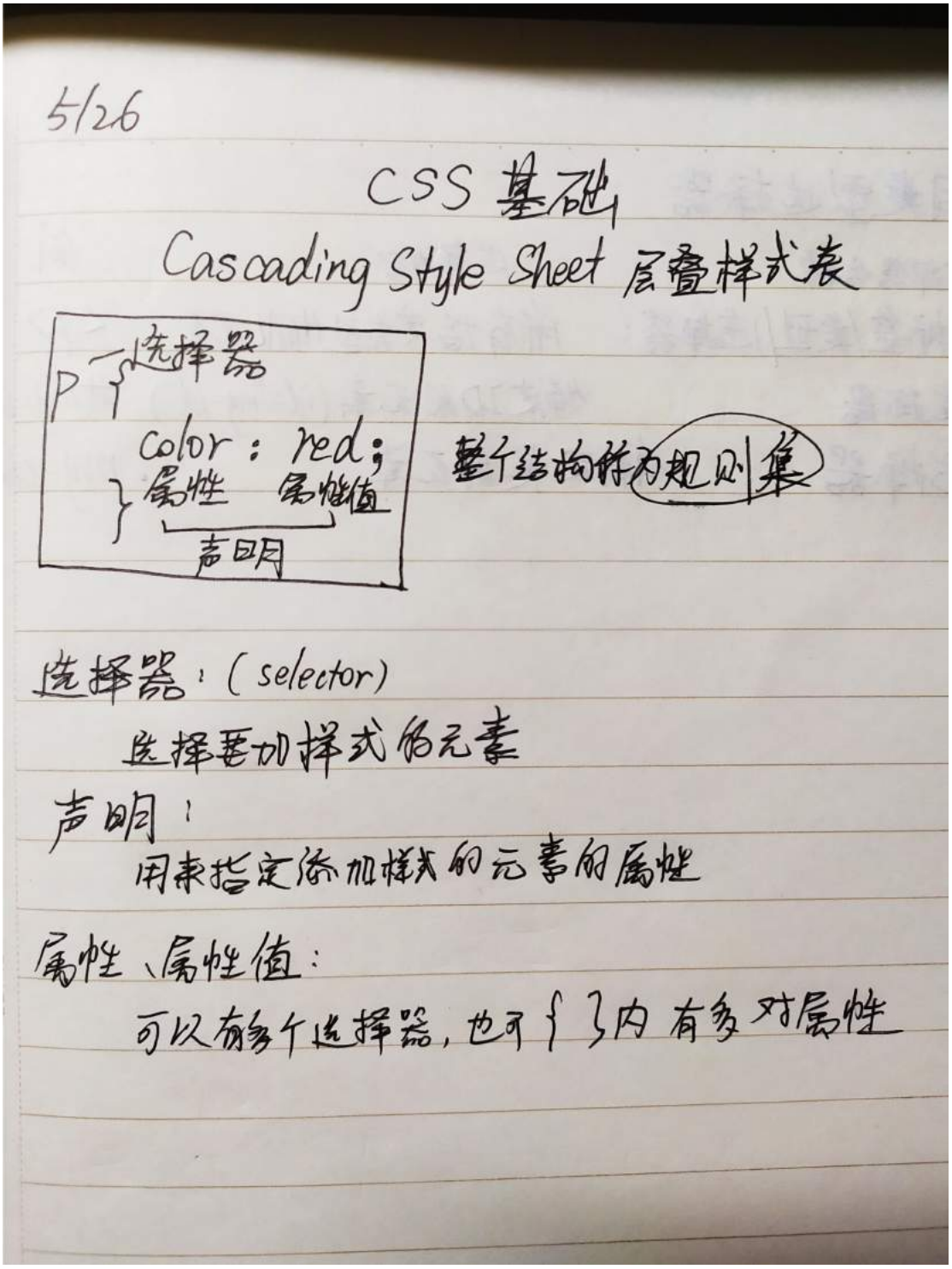
标题 <h>

段落 <p>

列表 < 无序
有序 <ol> 每个项目用 <li> 包围

链接 <a> anchor

例：这是链接



作者：张秋怡

链接：https://www.zhihu.com/question/22689579/answer/22318058

来源：知乎

一个普通网站访问的过程

简单概括一下，对于我们普通的网站访问，涉及到的技术就是：

用户操作浏览器访问，浏览器向服务器发出一个 HTTP 请求；

服务器接收到 HTTP 请求，Web Server 进行相应的初步处理，使用服务器脚本生成页面；

服务器脚本（利用 Web Framework）调用本地和客户端传来的数据，生成页面；

Web Server 将生成的页面作为 HTTP 响应的 body，根据不同的处理结果生成 HTTP header，发回给客户端；

客户端（浏览器）接收到 HTTP 响应，通常第一个请求得到的 HTTP 响应的 body 里是 HTML 代码，于是对 HTML 代码开始解析；

解析过程中遇到引用的服务器上的资源（额外的 CSS、JS 代码，图片、音视频，附件等），再向 Web Server 发送请求，Web Server 找到对应的文件，发送回来；

浏览器解析 HTML 包含的内容，用得到的 CSS 代码进行外观上的进一步渲染，JS 代码也可能会对外观进行一定的处理；

用户与页面交互（点击，悬停等等）时，JS 代码对此作出一定的反应，添加特效与动画；

交互的过程中可能需要向服务器索取或提交额外的数据（局部的刷新，类似微博的新消息通知），一般不是跳转就是通过 JS 代码（响应某个动作或者定时）向 Web Server 发送请求，Web Server 再用服务器脚本进行处理（生成资源 or 写入数据之类的），把资源返回给客户端，客户端用得到的资源来实现动态效果或其他改变。

注意这只是小网站里比较常见的模型，大网站为了解决规模问题还会有很多处理，每个环节都会有一些细微的差异，中间还会使用各种各样的工具减轻服务器的压力，提高效率，方便日常维护~